


# Statistical Data Analysis: Lecture 7

- 1 Probability, Bayes' theorem
- 2 Random variables and probability densities
- 3 Expectation values, error propagation
- 4 Catalogue of pdfs
- 5 The Monte Carlo method
- 6 Statistical tests: general concepts
-  7 **Test statistics, multivariate methods**
- 8 Goodness-of-fit tests
- 9 Parameter estimation, maximum likelihood
- 10 More maximum likelihood
- 11 Method of least squares
- 12 Interval estimation, setting limits
- 13 Nuisance parameters, systematic uncertainties
- 14 Examples of Bayesian approach

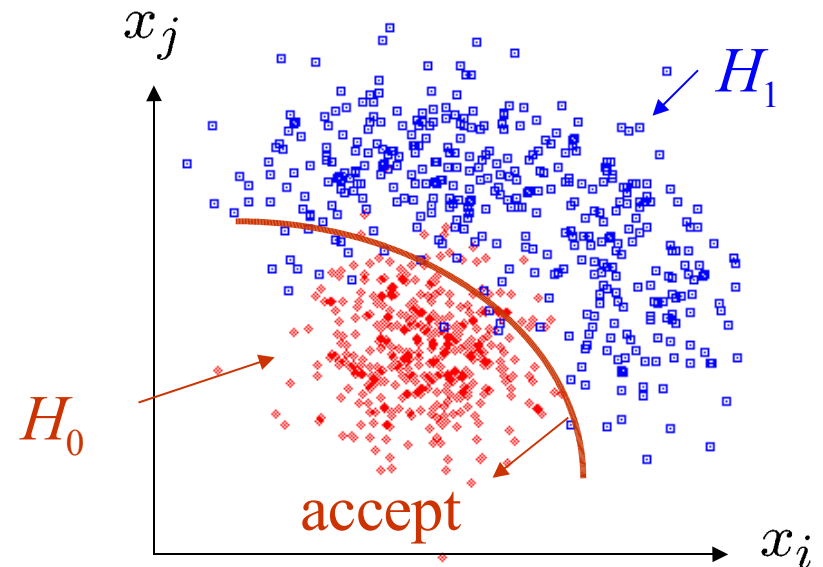
# Nonlinear test statistics

The optimal decision boundary may not be a hyperplane,

→ nonlinear test statistic  $t(\vec{x})$

Multivariate statistical methods  
are a Big Industry:

Neural Networks,  
Support Vector Machines,  
Kernel density methods,  
...



Particle Physics can benefit from progress in **Machine Learning**.

# Introduction to neural networks

Used in neurobiology, pattern recognition, financial forecasting, ...  
Here, neural nets are just a type of test statistic.

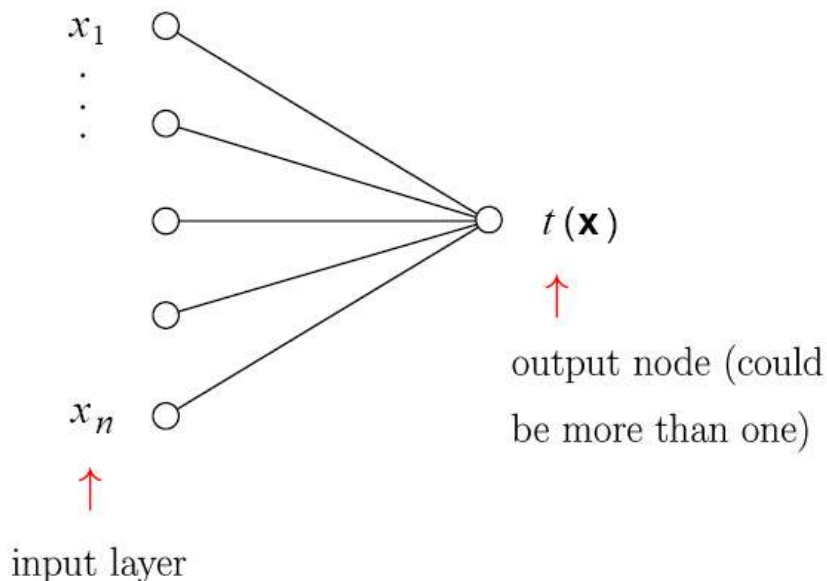
Suppose we take  $t(\mathbf{x})$  to have the form

$$t(\vec{x}) = s \left( a_0 + \sum_{i=1}^n a_i x_i \right), \text{ where } s(u) \equiv (1 - e^{-u})^{-1}.$$

logistic  
sigmoid

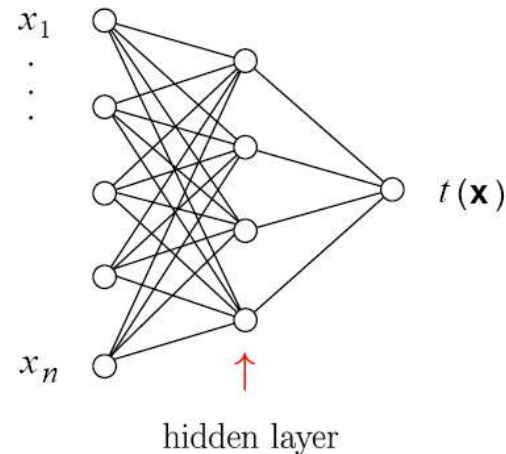
This is called the  
single-layer perceptron.

$s(\cdot)$  is monotonic  
→ equivalent to linear  $t(\mathbf{x})$



# The multi-layer perceptron

Generalize from one layer  
to the **multilayer perceptron**:



The values of the nodes in the  
intermediate (hidden) layer are

$$h_i(\vec{x}) = s \left( w_{i0} + \sum_{j=1}^n w_{ij}x_j \right) ,$$

and the network output is given by  $t(\vec{x}) = s \left( a_0 + \sum_{i=1}^n a_i h_i(\vec{x}) \right) .$

$a_i, w_{ij} =$  weights (connection strengths)

# Neural network discussion

Easy to generalize to arbitrary number of layers.

Feed-forward net: values of a node depend only on earlier layers, usually only on previous layer (“network architecture”).

More nodes  $\rightarrow$  neural net gets closer to optimal  $t(\mathbf{x})$ , but more parameters need to be determined.

Parameters usually determined by minimizing an error function,

$$\mathcal{E} = E_0[(t - t^{(0)})^2] + E_1[(t - t^{(1)})^2] ,$$

where  $t^{(0)}$ ,  $t^{(1)}$  are target values, e.g., 0 and 1 for logistic sigmoid.

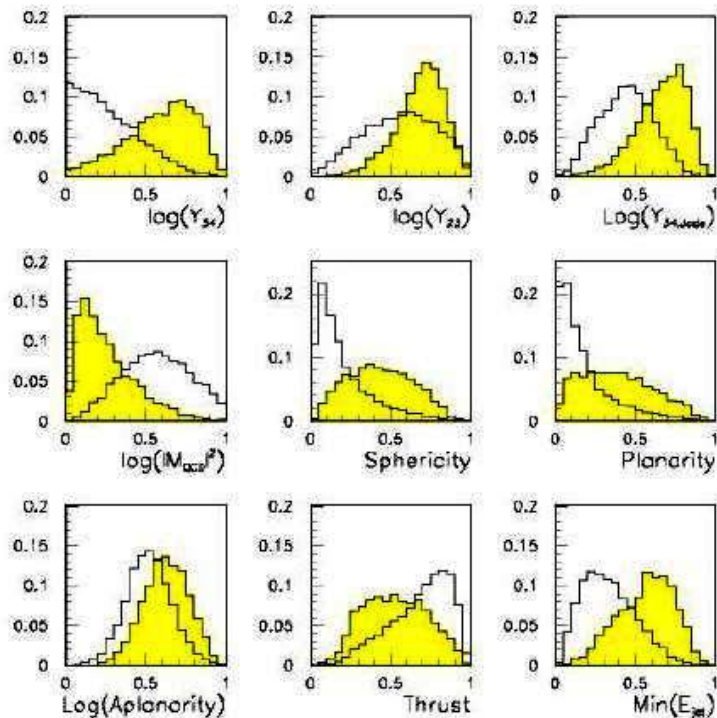
Expectation values replaced by averages of training data (e.g. MC).

In general training can be difficult; standard software available.

# Neural network example from LEP II

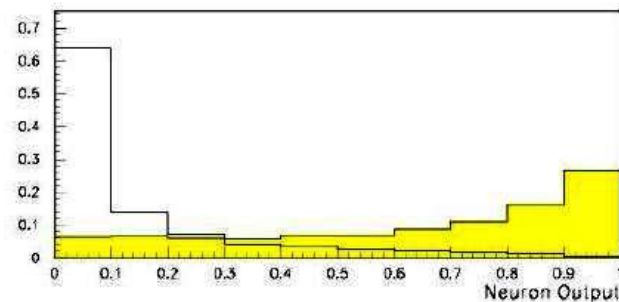
Signal:  $e^+e^- \rightarrow W^+W^-$  (often 4 well separated hadron jets)

Background:  $e^+e^- \rightarrow q\bar{q}g$  (4 less well separated hadron jets)



← input variables based on jet structure, event shape, ...  
none by itself gives much separation.

Neural network output does better...



(Garrido, Juste and Martinez, ALEPH 96-144)

# Probability Density Estimation (PDE) techniques

Construct non-parametric estimators of the pdfs  $f(\vec{x}|H_0)$ ,  $f(\vec{x}|H_1)$  :  
and use these to construct the likelihood ratio

$$t(\vec{x}) = \frac{\hat{f}(\vec{x}|H_0)}{\hat{f}(\vec{x}|H_1)}$$

( $n$ -dimensional histogram is a brute force example of this.)

More clever estimation techniques can get this to work for  
(somewhat) higher dimension.

See e.g. K. Cranmer, *Kernel Estimation in High Energy Physics*, CPC **136** (2001) 198; hep-ex/0011057;  
T. Carli and B. Koblitz, *A multi-variate discrimination technique based on range-searching*,  
NIM A **501** (2003) 576; hep-ex/0211019

# Kernel-based PDE (KDE, Parzen window)

Consider  $d$  dimensions,  $N$  training events,  $\mathbf{x}_1, \dots, \mathbf{x}_N$ ,  
estimate  $f(\mathbf{x})$  with

$$\hat{f}(\vec{x}) = \frac{1}{Nh^d} \sum_{i=1}^N K\left(\frac{\vec{x} - \vec{x}_i}{h}\right)$$

↑ kernel

bandwidth  
(smoothing parameter)

Use e.g. Gaussian kernel:  $K(\vec{x}) = \frac{1}{(2\pi)^{d/2}} e^{-|\vec{x}|^2/2}$

Need to sum  $N$  terms to evaluate function (slow);  
faster algorithms only count events in vicinity of  $\mathbf{x}$   
( $k$ -nearest neighbor, range search).



# Product of one-dimensional pdfs

First rotate to uncorrelated variables, i.e., find matrix  $A$  such that for  $\vec{x}' = A\vec{x}$  we have  $\text{cov}[x'_i, x'_j] = \delta_{ij}\sigma_i^2$ .

Estimate the  $d$ -dimensional joint pdf as the product of 1-d pdfs,

$$\hat{f}(\vec{x}) \approx \prod_{i=1}^d \hat{f}_i(x_i) \quad (\text{here } \mathbf{x} \text{ decorrelated})$$

This does not exploit non-linear features of the joint pdf, but simple and may be a good approximation in practical examples.

# Decision trees

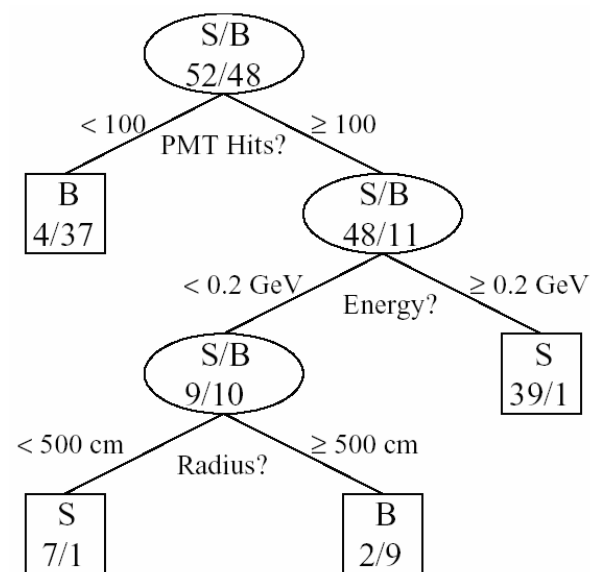
A training sample of signal and background data is repeatedly split by successive cuts on its input variables.

Order in which variables used based on best separation between signal and background.

Iterate until stop criterion reached, based e.g. on purity, minimum number of events in a node.

Resulting set of cuts is a ‘**decision tree**’.

Tends to be sensitive to fluctuations in training sample.



Example by Mini-Boone, B. Roe et al., NIM A **543** (2005) 577

# Boosted decision trees

Boosting combines a number classifiers into a stronger one; improves stability with respect to fluctuations in input data.

To use with decision trees, increase the weights of misclassified events and reconstruct the tree.

Iterate  $\rightarrow$  forest of trees (perhaps  $> 1000$ ). For the  $m$ th tree,

$$T_m(\vec{x}) = \begin{cases} 1 & \vec{x} \text{ in signal acceptance region} \\ -1 & \text{otherwise} \end{cases}$$

Define a score  $\alpha_m$  based on error rate of  $m$ th tree.

Boosted tree = weighted sum of the trees:  $T(\vec{x}) = \sum_m \alpha_m T_m(\vec{x})$

Algorithms: AdaBoost (Freund & Schapire),  $\epsilon$ -boost (Friedman).

# Multivariate analysis discussion

For all methods, need to check:

Sensitivity to statistically unimportant variables  
(best to drop those that don't provide discrimination);

Level of smoothness in decision boundary (sensitivity  
to over-training)

Given the test variable, next step is e.g., select  $n$  events and  
estimate a cross section of signal:  $\hat{\sigma}_s = (n - b)/\epsilon_s L$

Now need to estimate systematic error...

If e.g. training (MC) data  $\neq$  Nature, test variable is not optimal,  
but not necessarily biased.

But our estimates of background  $b$  and efficiencies would then  
be biased if based on MC. (True also for 'simple cuts'.)

## Multivariate analysis discussion (2)

But in a cut-based analysis it may be easier to avoid regions where untested features of MC are strongly influencing the decision boundary.

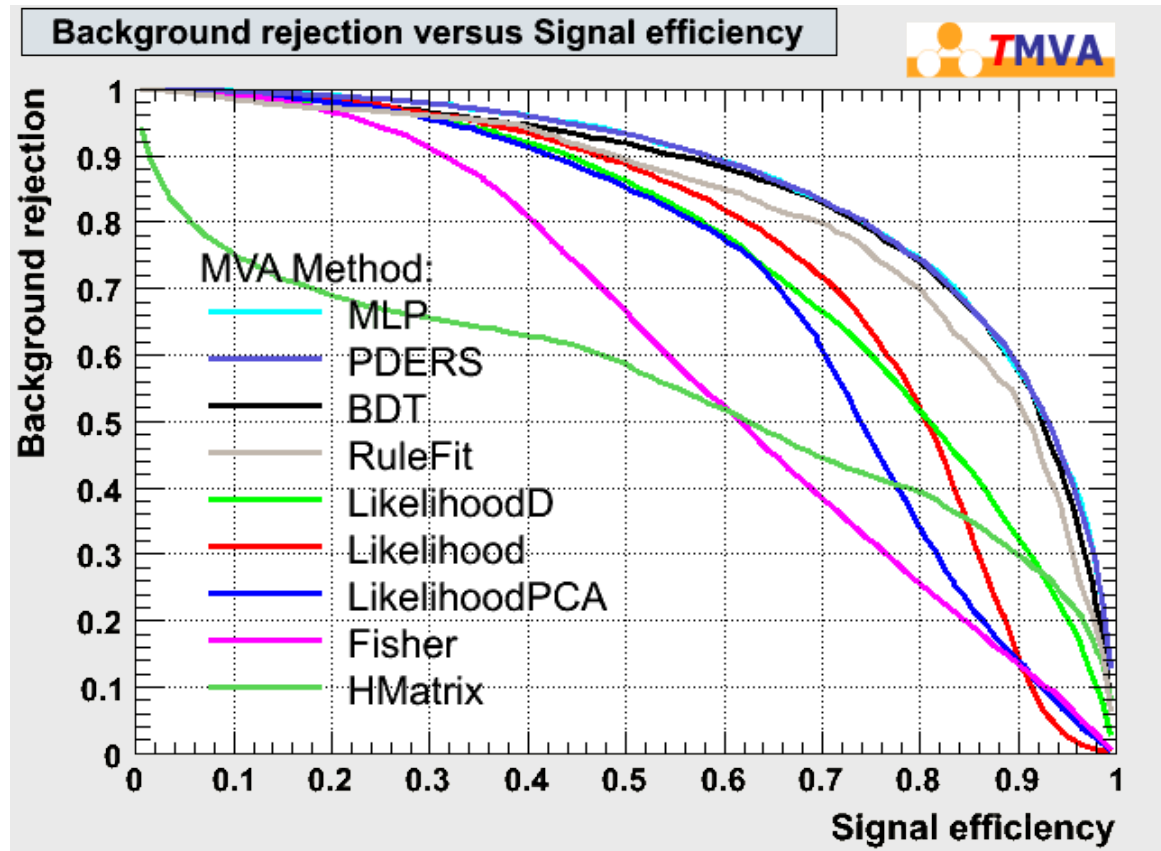
Look at control samples to test joint distributions of inputs.

Try to estimate backgrounds directly from the data (sidebands).

The purpose of the statistical test is often to select objects for further study and then measure their properties.

Need to avoid input variables that are correlated with the properties of the selected objects that you want to study. (Not always easy; correlations may be poorly known.)

# Comparing multivariate methods (TMVA)



Choose the best one!

# Some multivariate analysis references

Hastie, Tibshirani, Friedman, *The Elements of Statistical Learning*, Springer (2001);

Webb, *Statistical Pattern Recognition*, Wiley (2002);

Kuncheva, *Combining Pattern Classifiers*, Wiley (2004);

Specifically on neural networks:

L. Lönnblad et al., *Comp. Phys. Comm.*, 70 (1992) 167;

C. Peterson et al., *Comp. Phys. Comm.*, 81 (1994) 185;

C.M. Bishop, *Neural Networks for Pattern Recognition*, OUP (1995);

John Hertz et al., *Introduction to the Theory of Neural Computation*, Addison-Wesley, New York (1991).

# Wrapping up lecture 7

We looked at statistical tests and related issues:

discriminate between event types (hypotheses),  
determine selection efficiency, sample purity, etc.

Some modern (and less modern) methods were mentioned:

Fisher discriminants, neural networks,  
PDE, KDE, decision trees, ...

Next we will talk about goodness-of-fit tests:

$p$ -value expresses level of agreement between data  
and hypothesis